UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P O Box 1450
Alexandria, Virginia 22313-1450
www uspto gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/699,062	10/31/2003	Bryan M. Cantrill	03226/330001; SUN040156	2597

32615 7590 04/02/2008
OSHA LIANG L.L.P./SUN
1221 MCKINNEY, SUITE 2800
HOUSTON, TX 77010

EXAMINER	
MYINT, DENNIS Y	

ART UNIT	PAPER NUMBER
2162	

NOTIFICATION DATE	DELIVERY MODE
04/02/2008	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

lord@oshaliang.com
hernandez@oshaliang.com
DOCKETING@OSHALIANG.COM

Office Action Summary	**Application No.**	**Applicant(s)**	
	10/699,062	CANTRILL, BRYAN M.	
	Examiner	**Art Unit**	
	DENNIS MYINT	2162	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *12/10/2008*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3,12,14 and 24* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1, 3, 12, 14, and 24* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

DETAILED ACTION

1. This communication is responsive to Applicant's Amendment, filed on December 10, 2007.

2. Claims 1, 3, 12, 14, and 24 are currently pending in this application. Claims 1, 12, and 24 are independent claims. In the amendment filed on December 10, 2007, claims 2, 5-11, 13, and 15-23 were cancelled. Claims 1, 3, 12, 14, and 24 were cancelled. **This Office Action is made final.**

Claim Rejections - 35 USC § 103

3. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

4. Claims 1, 3, 12, and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Pang et al., (hereinafter "Pang", U.S. Patent Application Number 6493837) in view of Sharma et al., (hereinafter "Sharma", U.S. Patent Number 5511190).

As per claim 1, Pang is directed to a method for obtaining data from a kernel (Pang, Figure 3, i.e., ***Data received from calling thread?;*** Pang Column

1 Lines 16-21; Pang Column 1 Lines 31-37; and Pang Column 2 Lines 29-33) and teaches the limitations:

"obtaining, by a first processor, first data from the kernel using one of a plurality of probes in the kernel" (Pang Figure 2, i.e., *Processor n 200... Processor 2 200, Processor 1 200;* Figure 2, i.e., *Event Tracing Program 203;* Note that any of processor 1 ... processor n in Figure 2 of Pang could be either a first process or a second processor; Pang Column 4 Lines 60-67, i.e., *event tracing program 23* and *a data producer program 226* in combination maps to the probes in the kernel; Also note, Pang, Figure 3, i.e., ***Data received from calling thread?;*** Pang Column 1 Lines 16-21, i.e., *One well known method of* ***tracking system performance data*** *is to use counters, which are stored* ***values*** *that get incremented every time a particular event occurs. For example****, an operating system could provide one counter for keeping track of disk reads*** *on a particular drive and another counter for keeping track of disk writes;* Pang Column 1 Lines 31-37, i.e., ***Event tracing*** *is a technique often used in conjunction with counters to* ***track system performance data*** *by recording events of interest in a log buffer. For example, a read on disk number one can be recorded as "read #1." The total number of reads can then be tallied in a post-processing phase by analyzing the logged data. On multiprocessor systems, log buffers also tend to be global entities;* and Pang Column 2 Lines 29-33, i.e., *The invention is generally realized as an event tracing program.* ***The event tracing program generally receives performance data about an event occurring on the computer system*** *from a data producer program. The event tracing program* *responds by recording the event performance data in one of a set of a log buffers*);

(applying an aggregation function to the first data to obtain a first intermediate result);

"storing the first intermediate result in a first data set in a first aggregation buffer", (wherein the first data set comprises an aggregation identifier that identifies the aggregation function) (Pang Figure 2, i.e., *Log Buffer 204;* in the left bottom corner of Figure 2 of Prang, i.e., *204 Associated with processor 1, 204 associated with processor 2, ... 204 associated with processor n*);

"obtaining, by the first processor" (Pang Figure 2, i.e., *Processors 1 ... to... Processor n;* any one of those processors could be the first processor or second processor), "second data from the kernel using one of the plurality of probes" (Pang Figure 3, i.e., ***Data received from calling thread?;*** Particularly note the loop going back from step 312 Record data to log buffer to the beginning of the loop at 300 and then to 203 which determines "Data received with calling thread?"; As such, the first processor (or any processor) of Prang collects data from Kernel as many times as the loops run, which actually runs endlessly by way of *event tracing program 23* and *a data producer program 226*);

(applying the aggregation function to the second data and the first intermediate result to obtain a second intermediate result);

"(replacing) the first intermediate result (with) the second intermediate result in the first data set in the first aggregation buffer" (Figure 2, i.e., *Log Buffer 204;* Pang Column 4 Lines 60 through Column 5 lines 7, i.e., *The event tracing* *program 230 generally receives performance data about an event occurring on a computer system from a data producer program 226 which executes a calling thread on one of the processors 200 to request that event performance data be logged by the event tracing program 230. The data producer program 226 can be any application that owns the source of the event data. An example of a kernel-mode data producer program is an operating system, which would be able to detect when disk drives were accessed, when a user logon occurred, and the like. An example of a user-mode data producer is the MICROSOFT brand INTERNET INFORMATION SERVER, which can detect internet accesses, downloads, etc.* ***The event tracing program 230 responds by recording the event performance data in one of a set of a log buffers 204****;* As such, the first immediate result and second immediate result (or more than one intermediate results) are repeatedly stored in the log buffer 204 as depicted by the loop of Figure 3 of Pang);

"obtaining, by a second processor, third data from the kernel using one of the plurality of probes in the kernel" (Pang Figure 2, i.e., *Processor n 200… Processor 2 200, Processor 1 200;* Figure 2, i.e., *Event Tracing Program 203;* Note that any of processor 1 … processor n in Figure 2 of Pang could be either a first process or a second processor; Pang Column 4 Lines 60-67, i.e., *event tracing program 23* and *a data producer program 226* in combination maps to the probes in the kernel; Note that each log buffer is associated with one of a plurality process as depicted in Figure 2 of Pang, i.e., *;* in the left bottom corner of Figure 2 of Prang, i.e., *204 Associated with processor 1, 204 associated with* *processor 2, …. 204 associated with processor n;* As such, a third immediate result (or more than one intermediate results) are repeatedly stored in respective buffer log 204 associated with respective processor as depicted by the loop of Figure 3 of Pang *);*

(applying the aggregation function to the third data to obtain a third intermediate result);

"storing the third intermediate result in a second data set in a second aggregation buffer" (Figure 2 of Pang, i.e., *;* in the left bottom corner of Figure 2 of Prang, i.e., *204 Associated with processor 1, 204 associated with processor 2, …. 204 associated with processor n;* As such, a third immediate result (or more than one intermediate results) are repeatedly stored in respective buffer log 204 associated with respective processor as depicted by the loop of Figure 3 of Pang);

(wherein the second data set comprises the aggregation identifier);

"obtaining, by the second processor, fourth data from the kernel using one of the plurality of probes" (Pang Figure 2, i.e., *Processor n 200… Processor 2 200, Processor 1 200;* Figure 2, i.e., *Event Tracing Program 203;* Note that any of processor 1 … processor n in Figure 2 of Pang could be either a first process or a second processor; Pang Column 4 Lines 60-67, i.e., *event tracing program 23* and *a data producer program 226* in combination maps to the probes in the kernel; Note that each log buffer is associated with one of a plurality process as depicted in Figure 2 of Pang, i.e., *;* in the left bottom corner of Figure 2 of Prang, i.e., *204 Associated with processor 1, 204 associated with processor 2, …. 204* *associated with processor n;* As such, a fourth immediate result (or more than one intermediate results) are repeatedly stored in respective buffer log 204 associated with respective processor (a second process in this case) as depicted by the loop of Figure 3 of Pang);

(applying the aggregation function to the fourth data and the third intermediate result to obtain a fourth intermediate result);

"(replacing) the third intermediate result (with) the fourth intermediate result in the second data set in the second aggregation buffer" (Pang Figure 2, i.e., *Log Buffer 204;* Pang Column 4 Lines 60 through Column 5 lines 7, i.e., *The event tracing program 230 generally receives performance data about an event occurring on a computer system from a data producer program 226 which executes a calling thread on one of the processors 200 to request that event performance data be logged by the event tracing program 230. The data producer program 226 can be any application that owns the source of the event data. An example of a kernel-mode data producer program is an operating system, which would be able to detect when disk drives were accessed, when a user logon occurred, and the like. An example of a user-mode data producer is the MICROSOFT brand INTERNET INFORMATION SERVER, which can detect internet accesses, downloads, etc.* ***The event tracing program 230 responds by recording the event performance data in one of a set of a log buffers 204****;* As such, the third immediate result and fourth immediate result (or more than one intermediate results) are repeatedly stored in the log buffer 204 as depicted by the loop of Figure 3 of Pang);

"generating an aggregation result for the kernel (by applying the aggregation function to the third intermediate result and the fourth intermediate result)" (Pang Column 5 Lines 27-37, i.e., *The event tracing program 230 provides performance data to a consumer program 228, which can use the performance data in a variety of ways. The consumer program 228 may in turn provide the performance data or an analysis of it to a user via a graphical user interface. An example of a data consumer program is the PERFORMANCE MONITOR implemented on the MICROSOFT WINDOWS NT brand operating system*); and

"storing the aggregation result in a user-level buffer" (Pang Column 5 Lines 27-37, i.e., *The event tracing program 230 provides performance data to a consumer program 228, which can use the performance data in a variety of ways.* ***The consumer program 228 may in turn provide the performance data or an analysis of it to a user*** *via a graphical user interface. An example of a data consumer program is the PERFORMANCE MONITOR implemented on the MICROSOFT WINDOWS NT brand operating system*),

wherein the first aggregation buffer and the second aggregation buffer are kernel-level buffers" (Pang, Figure 2, i.e., Log Buffer 204, which is in Kernel level). Note that limitations in the parenthesis above are not explicitly taught by Pang but included in parentheses for the sake of easy reading.

Pang does not explicitly teach the following limitations: "applying an aggregation function to (the first data) to obtain a first intermediate result", "wherein (the first data set) comprises an aggregation identifier that identifies the aggregation function", "applying the aggregation function to (the second data) and (the first intermediate result) to obtain a second intermediate result", "applying the aggregation function to (the fourth data and the third intermediate result) to obtain a fourth intermediate result", and "by applying the aggregation function to (the third intermediate result and the fourth intermediate result)". Note that limitations in the parentheses are taught by Pang as discussed above but included of the sake of easy reading.

On the other hand, Sharma teaches the limitations:

"applying an aggregation function to the first data to obtain a first intermediate result" (Sharma, Column 2 Lines 44-47, i.e., *These hash-based techniques allow groupings and **aggregates to be generated** on the fly through the use of partial aggregates maintained in primary memory;* Sharma, Column 3 Lines 24-27, *the selected columns of database rows belonging to groups that can't fit into the group table **are buffered** then, when the buffer is full, written directly to an overflow disk file;* Also note Sharma Column 6 Lines 57-67, i.e., *The group table GT 218 consists of a number of group table entries, each summarizing aggregated database data for a single group. Note that a group can correspond to a value from a single group column or to a combination of values from multiple group columns. What is meant by "summarizing" depends on the particular grouping query being executed in the DBMS. If a user queried the DBMS for the maximum and minimum salaries for each department represented in the employee database, each group table entry would include min and max* *data fields in which the current minimum and maximum are retained as well as a department name field*);

"wherein (the first data set) comprises an aggregation identifier that identifies the aggregation function" (Sharma Column 2 Lines 59-63, i.e., *For each row, values are picked up for select columns designated in a SQL group-by statement, including* ***a group value or identifier*** *from the group columns, and zero or more data values from the data columns;* Sharma, Column 2 Line 67 through Column 4 Lines 4, i.e., *Each group table entry stores for a single group (i.e.,* ***a unique group identifier****) aggregates built on the group members' selected data fields, a group identifier, and housekeeping data;* Sharma, Column 7 Lines 40-49, i.e., *The largest possible size of the Group Table is determined by* ***the number of unique values of the group columns*** *GC 252;* (Particular note that unique group identifier corresponds to "aggregation identifier" of the claim invention because said unique group identifier identifies each grouping caused by "group by" (i.e., aggregating function)),

"applying the aggregation function to (the second data) and (the first intermediate result) to obtain a second intermediate result", "applying the aggregation function to (the fourth data and the third intermediate result) to obtain a fourth intermediate result", and "by applying the aggregation function to (the third intermediate result and the fourth intermediate result)" (Sharma, Column 2 Lines 44-47, i.e., *These hash-based techniques allow groupings and* ***aggregates to be generated*** *on the fly through the use of partial aggregates maintained in primary memory;* Sharma, Column 3 Lines 24-27, *the selected* *columns of database rows belonging to groups that can't fit into the group table* ***are buffered*** *then, when the buffer is full, written directly to an overflow disk file;* Also note Sharma Column 6 Lines 57-67, i.e., *The group table GT 218 consists of a number of group table entries, each summarizing aggregated database data for a single group. Note that a group can correspond to a value from a single group column or to a combination of values from multiple group columns. What is meant by "summarizing" depends on the particular grouping query being executed in the DBMS. If a user queried the DBMS for the maximum and minimum salaries for each department represented in the employee database, each group table entry would include min and max data fields in which the current minimum and maximum are retained as well as a department name field*).

At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the method of Pang to add the features of "applying an aggregation function to (the first data) to obtain a first intermediate result", "wherein (the first data set) comprises an aggregation identifier that identifies the aggregation function", "applying the aggregation function to (the second data) and (the first intermediate result) to obtain a second intermediate result", "applying the aggregation function to (the fourth data and the third intermediate result) to obtain a fourth intermediate result", and "by applying the aggregation function to (the third intermediate result and the fourth intermediate result)", as taught by Sharma, to the method of Pang so that the resultant method would applying an aggregation function to the first data to obtain a first intermediate result", the first data set would comprise an aggregation identifier that identifies the aggregation function, the resultant method would apply the aggregation function to the second data and the first intermediate result to obtain a second intermediate result, the resultant method would apply the aggregation function to the fourth data and the third intermediate result to obtain a fourth intermediate result" and the resultant method would apply the aggregation function to the third intermediate result and the fourth intermediate result. One would have been motivated to do so because *there is a need for a grouping or aggregation process that, through efficient use of computer memory resources, can run largely in memory, yielding increased execution speeds* (Sharma, Column 2 Lines 22-25).

As per claim 3, Pang in view of Sharma teaches the limitations:

"obtaining an expression" (Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function applies* ***the hash function HF 210*** *to the group identifier (i.e., "A10") associated with the first row. The resulting hashed group value (HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Note that a hash function is an expression), "the first intermediate result" (Pang Figure 2, i.e., *Log Buffer 204;* in the left bottom corner of Figure 2 of Prang, i.e., *204 Associated with processor 1, 204 associated with processor 2, …. 204 associated with processor n*), and "the aggregation identifier" (Sharma Column 2 Lines 59-63, i.e., *For each row, values are picked up for select columns* *designated in a SQL group-by statement, including* ***a group value or identifier*** *from the group columns, and zero or more data values from the data columns;* Sharma, Column 2 Line 67 through Column 4 Lines 4, i.e., *Each group table entry stores for a single group (i.e.,* ***a unique group identifier****) aggregates built on the group members' selected data fields, a group identifier, and housekeeping data;* Sharma, Column 7 Lines 40-49, i.e., *The largest possible size of the Group Table is determined by* ***the number of unique values of the group columns*** *GC 252;* (Particular note that unique group identifier corresponds to “aggregation identifier” of the claim invention because said unique group identifier identifies each grouping caused by “group by” (i.e., aggregating function)) , and

“generating a key using the expression and the aggregation identifier” (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier)), "wherein the key is used to locate the first data set" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier); Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang) and "wherein the first data set comprises the key" (Pang Figure 2, i.e., *Offset Variable 206;* Pang Column 6 Lines 1-5, i.e., *an offset value variable 206;* Note that the offset variable 206 in the log buffer of Pang holds a pointer to the data set which is the log buffer itself (first or second dataset or the like). Sharma hashed key is also a pointer into a data object/set. As such, Pang in view of Sharma would teach a "first data set" which comprises the key").

As per claim 12, Pang in view of Sharma teaches the limitations:

"obtaining a first data set from a first aggregation buffer" (Pang in view of Sharma, that is, Pang Figure 2, i.e., *Offset Variable 206;* Note that "Offset Variable 206" of Pang maps to a first data set from a first aggregation buffer. Figure 2, i.e., *Log Buffer 204;* Pang Column 4 Lines 60 through Column 5 lines 7, i.e., *The event tracing program 230 generally receives performance data about an event occurring on a computer system from a data producer program 226 which executes a calling thread on one of the processors 200 to request that event performance data be logged by the event tracing program 230. The data producer program 226 can be any application that owns the source of the event data. An example of a kernel-mode data producer program is an operating system, which would be able to detect when disk drives were accessed, when a user logon occurred, and the like. An example of a user-mode data producer is the MICROSOFT brand INTERNET INFORMATION SERVER, which can detect internet accesses, downloads, etc.* ***The event tracing program 230 responds by recording the event performance data in one of a set of a log buffers 204****;* As such, the first immediate result and second immediate result (or more than one intermediate results) are repeatedly stored in the log buffer 204 as depicted by the loop of Figure 3 of Pang. In addition, as Figure 2 of Pang depicts, there are more than one log buffer generated by more than one CPU. As such, there are a first log buffer and a first data set contained the first log buffer, a second log buffer and a second data set contained in the second log buffer and so on as long as the loop of Pang as depicted in Figure 3 of Pang keeps on repeating), " wherein the first data set comprises a key component" (Pang Figure 2, i.e., *Offset Variable 206;* Pang Column 6 Lines 1-5, i.e., *an offset value variable 206;* Note that the offset variable 206 in the log buffer of Pang holds a pointer to the data set which is the log buffer itself (first or second dataset or the like). Sharma hashed key is also a pointer into a data object/set. As such, Pang in view of Sharma would teach a "first data set" which comprises the key") , "an aggregation identifier component" (Sharma Column 2 Lines 59-63, i.e., *For each row, values are picked up for select columns designated in a SQL group-by statement, including* ***a group value or identifier*** *from the group columns, and zero or more data values from the data columns;* Sharma, Column 2 Line 67 through Column 4 Lines 4, i.e., *Each group table entry stores for a single group (i.e.,* ***a unique group identifier****) aggregates built on the group members' selected data fields, a group identifier, and housekeeping data;* Sharma, Column 7 Lines 40-49, i.e., *The largest possible size of the Group Table is determined by* ***the number of unique values of the group columns*** *GC 252;* (Particular note that unique group identifier corresponds to "aggregation identifier" of the claim invention because said unique group identifier identifies each grouping caused by "group by" (i.e., aggregating function)), and "a value component" (Sharma Column 2 Lines 59-63, i.e., *For each row, values are picked up for select columns designated in a SQL group-by statement, including a group value or identifier from the group columns, and* ***zero or more data values from the data columns;*** Sharma, Column 6 Lines 37-31, i.e., *At least one of the selected columns SC 250 is designated as a group column GC 252.* ***The remaining (zero***

or more) selected columns SC 250 are data columns DC 254, which provide the member data to be grouped or aggregated), "wherein the value component comprises a first intermediate aggregation result obtained by applying an aggregation function to data obtained during execution of a first processor" (In the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored. Note Sharma, Column 2 Lines 44-47, i.e., *These hash-based techniques allow groupings and* ***aggregates to be generated*** *on the fly through the use of partial aggregates maintained in primary memory;* Sharma, Column 3 Lines 24-27, *the selected columns of database rows belonging to groups that can't fit into the group table* ***are buffered*** *then, when the buffer is full, written directly to an overflow disk file;* Also note Sharma Column 6 Lines 57-67, i.e., *The group table GT 218 consists of a number of group table entries, each summarizing aggregated database data for a single group. Note that a group can correspond to a value from a single group column or to a combination of values from multiple group columns. What is meant by "summarizing" depends on the particular grouping query being executed in the DBMS. If a user queried the DBMS for the maximum and minimum salaries for each department represented in the employee database, each group table entry would include min and max data fields in which the current minimum and maximum are retained as well as a department name field*),

"wherein the aggregation identifier component comprises an aggregation identifier that identifies the aggregation function" (Sharma Column 2 Lines 59-63, i.e., *For each row, values are picked up for select columns designated in a SQL* *group-by statement, including* ***a group value or identifier*** *from the group columns, and zero or more data values from the data columns;* Sharma, Column 2 Line 67 through Column 4 Lines 4, i.e., *Each group table entry stores for a single group (i.e.,* ***a unique group identifier****) aggregates built on the group members' selected data fields, a group identifier, and housekeeping data;* Sharma, Column 7 Lines 40-49, i.e., *The largest possible size of the Group Table is determined by* ***the number of unique values of the group columns*** *GC 252;* (Particular note that unique group identifier corresponds to "aggregation identifier" of the claim invention because said unique group identifier identifies each grouping caused by "group by" (i.e., aggregating function)) , and

"wherein the first aggregation buffer is a kernel-level buffer associated with the first processor" (As Figure 2 of Pang depicts, there are more than one log buffer generated by more than one CPU. As such, there are a first log buffer and a first data set contained the first log buffer, a second log buffer and a second data set contained in the second log buffer and so on as long as the loop of Pang as depicted in Figure 3 of Pang keeps on repeating. Pang, Figure 2, i.e., Log Buffer 204, which is in Kernel level);

"obtaining the aggregation identifier and using the aggregation identifier to obtain a user-level table key" (Sharma Column 2 Lines 59-63, i.e., *For each row, values are picked up for select columns designated in a SQL group-by statement, including* ***a group value or identifier*** *from the group columns, and zero or more data values from the data columns;* Sharma, Column 2 Line 67 through Column 4 Lines 4, i.e., *Each group table entry stores for a single group* *(i.e., **a unique group identifier**) aggregates built on the group members' selected data fields, a group identifier, and housekeeping data;* Sharma, Column 7 Lines 40-49, i.e., *The largest possible size of the Group Table is determined by **the number of unique values of the group columns** GC 252;* (Particular note that unique group identifier corresponds to "aggregation identifier" of the claim invention because said unique group identifier identifies each grouping caused by "group by" (i.e., aggregating function); The table of Sharma is at user-level. As such, the table key of Sharma (hashed value) is the user-level table key);

"hashing the user-level table key to obtain a generated hash key" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure **applies a hash function to the group identifier, generating a hashed group value** that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function **applies the hash function HF 210 to the group identifier** (i.e., "A10") associated with the first row. The resulting **hashed group value** (HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier));

"locating a user-level table entry, in the user-level table comprising a hash key matching the generated hash key" (Sharma, Column 2 Lines 63-67, i.e., *a* *matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier)); and

"updating a current value of a value component of the user-level table entry to obtain a new value" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see

Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier); Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang which would be stored in Sharma's table. As the loop of Pang in view of Sharma continues repeating, the table of Sharma would continuously updated to obtain a new value because in the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored),

"werein updating the current value component comprises applying the aggregation function to the current value and the first intermediate aggregation result to obtain the new value" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier); Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang which would be stored in Sharma's table. As the loop of Pang in view of Sharma continues repeating, the table of Sharma would continuously updated to obtain a new value because in the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored),

"herein the current value is a second intermediate result obtained by applying the aggregation function to data obtained during execution of a second processor" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier); Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang which would be stored in Sharma's table. As the loop of Pang in view of Sharma continues repeating, the table of Sharma would continuously updated to obtain a new value because in the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored),

"werein the second intermediate result is stored in a second data set comprising the aggregation identifier" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier); Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang which would be stored in Sharma's table. As the loop of Pang in view of Sharma continues repeating, the table of Sharma would continuously updated to obtain a new value because in the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored)

"herein the second intermediate result is stored in a second aggregation buffer prior to being stored in the value component" (Sharma, Column 2 Lines 63-67, i.e., *a matching procedure* ***applies a hash function to the group identifier, generating a hashed group value*** *that serves as an index into a memory-resident hash table that maps hashed group values into corresponding memory-resident group table entries;* Sharma, Column 9 Lines 31-37, i.e., *After the first row is read, the grounding function* ***applies the hash function HF 210 to the group identifier*** *(i.e., "A10") associated with the first row. The resulting* ***hashed group value*** *(HF(A10)) serves as an index to an entry (HT[HF(A10)]) of the hash table HT 216, the contents of which contain the address (*GT[A10]) of the group table entry (if one exists) in which data for group A10 is being aggregated;* Also see Figure 2 of Sharma; Note that a key (i.e., a hashed group value) is generated using the expression (a hash function) and the aggregation identifier (i.e., group identifier); Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang which would be stored in Sharma's table. As the loop of Pang in view of Sharma continues repeating, the table of Sharma would continuously updated to obtain a new value because in the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored), "wherein the second aggregation buffer is a kernel-level buffer associated with the second processor" (Pang, Figure 2, i.e., Log Buffer 204, which is in Kernel level. Note that Pang teaches more than one log buffer and more than one CPU).

Claim 24 is essentially the same as claim 1 except that it set forth the claimed invention as a computer system on a network obtaining data from a kernel rather than a method for obtaining data from a kernel and rejected for the same reasons as applied hereinabove.

5. Claims14 is rejected under 35 U.S.C. 103(a) as being unpatentable over Pang in view of Sharma and further in view of Barnett (U.S. Patent Application Publication Number 2003/0159132).

Referring to claim 14, Pang in view of Sharma does not explicitly recite the use of dictionaries. However, using data dictionaries along with hash tables is well known in the art. For instance, Barnett teaches a method and system for conformance checking, wherein data dictionaries are used along with hash tables (Barnett et al. Paragraph 0028 and 0040).

As such, Pang in view of Sharma and further in view of Barnett teaches the limitation:

"wherein obtaining the aggregation identifier comprises searching at least one selected from the group consisting of a user-level dictionary and a kernel-level dictionary" (Pang in view of Sharma and further in view of Barnett et al. Paragraph 0028 and 0040).

At the time the invention was made, it would have been obvious to a person of ordinary skill in the art to modify the method of Pang in view of Sharma and further in view of Larson to add the feature of using data dictionaries along with hash tables as taught by Barnett to the method of Pang in view of Sharma and further in view of Larson so that in resultant method and system would be directed to the method claim 12, wherein obtaining the aggregation identifier matching the value of the aggregation identifier comprises search at least one selected from the group consisting of a user-level dictionary and a kernel level dictionary. Note that Tang et al. teaches data from user-level and kernel-level. One would have been motivated to do so in order to perform retrieval operations in a hash method, wherein key-value pairs are collected together as a dictionary (Barnett, Paragraph 0040).

Response to Arguments

6. The applicant's arguments filed on December 10, 2007 have been fully considered but are not persuasive.

Applicant argued that "none of the cited prior art teaches or suggests all of the aforementioned limitations. Specifically, the Examiner admits that Pang does not teach aggregating data" (Applicant's argument, page 9 second paragraph). Applicant also argued that "Sharma does not teach or suggest: (i) aggregation of data on a per-processor basis to obtain intermediate results, (ii) storing intermediate results in a kernel-level buffer, (iii) storing aggregation identifiers with the intermediate results; and (iv) aggregating data in a user-level buffer" (Applicant's argument, page 9 second paragraph).

In response it is pointed out that said limitations are rejected over the combination of Pang in view of Sharma. Note Pang Figure 3, i.e., ***Data received from calling thread?*** Particularly note the loop going back from step 312 Record data to log buffer to the beginning of the loop at 300 and then to 203 which determines "Data received with calling thread?"; As such, the first processor (or any other processor) of Prang collects data from respective Kernel as many times as the loop runs, which actually runs endlessly by way of *event tracing program 23* and *a data producer program 226.* Also note that Note that in the method of Pang in view of Sharma, the key generated by Sharma's method of generating a hashed group value would identify the first data set of Pang which would be stored in Sharma's table. As the loop of Pang in view of Sharma continues repeating, the table of Sharma would continuously updated to obtain a new value because in the method of Pang in view of Sharma, log buffers of Pang would be stored in Sharma's table wherein value components are stored. Therefore in the method of Pang in view of Sharma, data would continuously aggregated/updated and stored in Sharma's table which is in the uer-level (Sharma, Column 2 Lines 44-47, i.e., *These hash-based techniques allow groupings and* ***aggregates to be generated*** *on the fly through the use of partial aggregates maintained in primary memory;* Sharma, Column 3 Lines 24-27, *the selected columns of database rows belonging to groups that can't fit into the* *group table* ***are buffered*** *then, when the buffer is full, written directly to an overflow disk file;* Also note Sharma Column 6 Lines 57-67, i.e., *The group table GT 218 consists of a number of group table entries, each summarizing aggregated database data for a single group. Note that a group can correspond to a value from a single group column or to a combination of values from multiple group columns. What is meant by "summarizing" depends on the particular grouping query being executed in the DBMS. If a user queried the DBMS for the maximum and minimum salaries for each department represented in the employee database, each group table entry would include min and max data fields in which the current minimum and maximum are retained as well as a department name field).*

As per applicant's argument that "*Sharma does not teach or suggest (i) aggregation of data on a per-processor basis to obtain intermediate results, (ii) storing intermediate results in a kernel-level buffer, (iii) storing aggregation identifiers with the intermediate results; and (iv) aggregating data in a user-level buffer*", it is pointed out that Pang in view of Sharma teaches *(i) aggregation of data on a per-processor basis to obtain intermediate results* (the method of Pang in view of Sharma comprises more than one CPUs with respective log buffers, wherein intermediate results as stored in the log buffers of Pang are later stored in Sharma's table, as discussed in details above; as such, Pang in view of Sharma teaches aggregation of data on a per-processor basis to obtain intermediate results), *(ii) storing intermediate results in a kernel-level buffer* (As explained above with respect to claim a, Pang teaches storing intermediate results in a respective log buffer (Pang Figure 2) in the kernel-level), *(iii) storing aggregation identifiers with the intermediate results* (in the method of Pang in view of Sharma, aggregation identifiers are stored in the user-level table of Sharma along with the intermediate results (stored in the log buffers of Pang), and *(iv) aggregating data in a user-level buffer* (data is repeatedly aggregated based on the loop of Figure 3 of Pang in Sharma's user-level table).

Referring to claims 5, 7-9, 12, 13, 15, 20, and 24, Applicant argued that "*as discussed above, Pang and Sharma fail to teach or suggest all the elements of the amended independent claims. Further, Larson fail to teach or suggest that as evidenced by the fact that Larson is only relied upon to teach concepts related to hashing*" (Applicant's argument, page 10 second paragraph).

In response, Applicant is reminded, in light of the cancellation of the claims, the Larson reference is not applied in the current office action and said argument is moot.

Referring to claims 14, 19, and 20-23, Applicant argued that "*as discussed above, Pang and Sharma fail to teach or suggest all the elements of the amended independent claims. Further, Barnett fail to teach or suggest that as evidenced by the fact that Larson is only relied upon to teach concepts related to hashing*" (Applicant's argument, page 10 last paragraph).

In response, it is pointed out that, in light of the cancellation of the claims, only claim 14 is relevant with respect to this argument. As per claim 14, it is pointed out that Pang in view of Sharma teaches each and every limitation of claim 12 which claim 14 depends upon and Burnett teaches what Pang in view of Sharma does not explicitly teaches as discussed above. As such, Applicant's argument is moot.

In view of the above, the examiner contends that all limitations as recited in the claims have been addressed in this Office Action. For the above reasons, Examiner believed that rejections of the Last Office action and Current Office Action are proper.

Conclusion

7. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Contact Information

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Dennis Myint whose telephone number is (571) 272-5629. The examiner can normally be reached on 8:30AM-5:30PM Monday-Friday.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, John Breene can be reached on (571) 272-4107. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Cam Y Truong/

Primary Examiner, Art Unit 2162

Dennis Myint
Examiner
AU-2162